



500.42885X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: T. TANAKA, et al

Serial No.: 10/606,955

Filing Date: June 27, 2003

PROGRAM GENERATION METHOD



**LETTER CLAIMING RIGHT OF PRIORITY**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

October 30, 2003

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, applicants hereby claim  
the right of priority based on:

**Japanese Application No. 2002-188938**  
**Filed: June 28, 2002**

A Certified copy of said application document is attached hereto.

Respectfully submitted,

\_\_\_\_\_  
Carl I. Brundidge  
Registration No. 29,621  
ANTONELLI, TERRY, STOUT & KRAUS, LLP

CIB/jdc  
Enclosures  
703/312-6600

日 本 国 特 許 庁  
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日            2 0 0 2 年   6 月 2 8 日  
Date of Application:

出 願 番 号            特 願 2 0 0 2 - 1 8 8 9 3 8  
Application Number:  
[ST. 10/C]:            [ J P 2 0 0 2 - 1 8 8 9 3 8 ]

出      願      人            株式会社日立製作所  
Applicant(s):

2 0 0 3 年   7 月   9 日

特許庁長官  
Commissioner,  
Japan Patent Office

太田信一郎

出証番号   出証特 2 0 0 3 - 3 0 5 4 3 5 4

【書類名】 特許願

【整理番号】 K02008741A

【あて先】 特許庁長官殿

【国際特許分類】 G06F 11/28

【発明者】

【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

【氏名】 田中 匠

【発明者】

【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

【氏名】 森崎 謙一

【特許出願人】

【識別番号】 000005108

【氏名又は名称】 株式会社日立製作所

【代理人】

【識別番号】 100075096

【弁理士】

【氏名又は名称】 作田 康夫

【手数料の表示】

【予納台帳番号】 013088

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム生成方法及びその実行処理方法並びにその実施装置

【特許請求の範囲】

【請求項 1】

画面を表示もしくは表示制御するための言語で記述された画面ファイルを入力し、該画面ファイルに基づいて、該画面ファイルからデータ取得するパラメータを入力として所定の業務を実行するための業務プログラムの雛型を生成することを特徴とするプログラム生成方法。

【請求項 2】

上記画面ファイルを作成する際、該画面フォームでデータ取得するデータの型または属性を定義するステップと、上記業務プログラムを呼出す関数名を指定するステップと、上記業務プログラムが記述されたプログラム言語を指定するステップを有することを特徴とする請求項1記載のプログラム生成方法。

【請求項 3】

請求項 2 記載のプログラム生成方法において、

上記業務プログラムを呼出す際に、予め決められたフォーマットのデータ群から上記業務プログラムを呼出すために必要なデータを抽出するステップと、上記業務プログラムが記述されたプログラム言語のデータ定義に変換するステップとを有することを特徴とするプログラム実行方法。

【請求項 4】

画面を表示もしくは表示制御するための言語で記述された画面ファイルを入力する手段と、該画面ファイルに基づいて、該画面ファイルからデータ取得するパラメータを入力として所定の業務を実行するための業務プログラムの雛型を生成する手段とを備えたことを特徴とするプログラム生成装置。

【請求項 5】

上記画面ファイルを作成する際、該画面フォームでデータ取得するデータの型または属性を定義する手段と、上記業務プログラムを呼出す関数名を指定する手段と、上記業務プログラムが記述されたプログラム言語を指定する手段を備えたことを特徴とする請求項1記載のプログラム生成装置。

**【請求項 6】**

請求項 2 記載のプログラム生成方法において、

上記業務プログラムを呼出す際に、予め決められたフォーマットのデータ群から上記業務プログラムを呼出すために必要なデータを抽出する手段と、上記業務プログラムが記述されたプログラム言語のデータ定義に変換する手段とを備えたことを特徴とするプログラム実行装置。

**【発明の詳細な説明】****【0 0 0 1】****【発明の属する技術分野】**

本発明はブラウザを通じた画面入力から、画面に連動したプログラムを生成または実行するためのプログラム生成技術やプログラム実行処理技術に関する。

**【0 0 0 2】****【従来の技術】**

画面表示言語で記述されたファイルから取得した入力情報をサーバ側で処理するシステム開発において、入力パラメータは、全て統一フォーマットの一括送信という形が広く普及している。その為、パラメータを分割し、必要なデータを取り出す処理をテンプレート化して、画面情報を元に、業務アプリケーションのインタフェースを作成する手法が知られている。

**【0 0 0 3】****【発明が解決しようとする課題】**

上記パラメータを分割するところまでは可能であるが、分割された個々のデータを様々な言語でコーディングされた業務アプリケーションと連携させるためには、各言語特有のノウハウが必要となる。画面を表示するプログラムと親和性のある言語の場合はあまり問題ではないが、画面を表示する機能が少ないもしくは機能を有さないプログラム言語（COBOL等のような）で記述されたプログラムと連携する場合、インタフェースが作成されても、それを利用した場合、プログラムの開発効率や再利用性が低下してしまうという課題があった。

**【0 0 0 4】**

本発明の目的は、上述の従来型における課題に鑑み、プログラム開発者が所望

するプログラム言語でコーディングされた業務プログラムを容易に連携できるように上記プログラム言語でコーディングされた業務プログラムの雛型を作成することを可能とするプログラム生成方法およびその実行処理方法並びにその実施装置を提供することにある。

#### 【 0 0 0 5 】

##### 【課題を解決するための手段】

上記目的を達成するため、請求項 1 が関わる発明は、画面ファイルを作成することで業務プログラムの雛型を自動生成することができるプログラム生成方法を特徴とする。

#### 【 0 0 0 6 】

請求項 2 が関わる発明は、請求項 1 で述べたプログラム生成方法を実装するために、画面ファイル上で、入力されるデータを定義するステップと呼出す関数名を定義するステップと開発言語を指定するステップを備えることを特徴とする。

#### 【 0 0 0 7 】

請求項 3 が関わる発明は、連結されたパラメータから必要なデータを抽出するステップと、それぞれの言語にそったデータ定義に変換するステップを備えたことを特徴とする。

#### 【 0 0 0 8 】

##### 【発明の実施の形態】

以下、図面を用いて、本発明の実施の形態を説明する。

図 1 にプログラム呼出し部及び生成されたプログラムの処理の実施例を示す。端末 1 0 0 上で、ユーザからの入力、または業務プログラムからのデータの参照など、データの入出力を必要とする画面が画面 A 1 1 0、画面 B 1 2 0、画面 C 1 3 0 と遷移する。ネットワーク 1 4 0 を介して、それぞれの画面が、Webサーバ 1 5 0 にアクセスする。画面からのデータはリクエストという形で、送信される。リクエストの中身は、ユーザからの入力データのほかに、データ送信の際に必要な様々なデータが、連結された状態になっている。Servlet 1 6 0 は画面からの入力データをそれぞれに対応したプログラム呼出し部に転送する役割を持つ。Native インタフェース 1 7 0 及び 1 8 0 は COBOL または Java (Java は S u n M

icrosystems, Inc. の商標もしくは登録商標である) に対応したインタフェースであり、データの設定や取得の実装の言語依存部分を隠蔽した機能を持つ。

#### 【0009】

画面 A 1 1 0 のアクセスをServlet 1 6 0 が受け取り、リクエストを画面 A 1 1 0 に対応したプログラム呼出部 A 1 1 1 に渡す。プログラム呼出部 A 1 1 1 はリクエストの中から、COBOL業務プログラム A 1 1 2 に渡す必要のあるデータを抽出する。抽出されたデータはCOBOL業務プログラム A 1 1 2 のパラメータとして指定されているデータ型に変換される。プログラム呼出し部 A 1 1 1 は変換の終わったデータをNativeインタフェース (COBOL) 1 7 0 を介してCOBOL業務プログラム A 1 1 2 に設定する。最後に、COBOL業務プログラム A 1 1 2 を実行させる。

#### 【0010】

画面 B 1 2 0 のアクセスはServlet 1 6 0 が受け取り、リクエストを画面 B 1 2 0 に対応したプログラム呼出部 B 1 2 1 に渡す。プログラム呼出部 B 1 2 1 はリクエストの中から、COBOL業務プログラム B 1 2 2 に渡す必要のあるデータを抽出する。抽出されたデータはCOBOL業務プログラム B 1 2 2 のパラメータとして指定されているデータ型に変換される。プログラム呼出し部 B 1 2 1 は変換の終わったデータをNativeインタフェース (COBOL) 1 7 0 を介してCOBOL業務プログラム A 1 2 2 に設定する。最後に、COBOL業務プログラム B 1 2 2 を実行させる。

#### 【0011】

画面 C 1 3 0 のアクセスをServlet 1 6 0 が受け取り、リクエストを画面 C 1 3 0 に対応したプログラム呼出部 C 1 3 1 に渡す。プログラム呼出部 C 1 3 1 はリクエストの中から、Java業務プログラム C 1 3 2 に渡す必要のあるデータを抽出する。抽出されたデータはJava業務プログラム C 1 3 2 のパラメータとして指定されているデータ型に変換される。プログラム呼出し部 C 1 3 1 は変換の終わったデータをNativeインタフェース (Java) 1 8 0 を介してJava業務プログラム C 1 3 2 に設定する。最後に、Java業務プログラム C 1 3 2 を実行させる。

#### 【0012】

それぞれの業務プログラムは必要に応じてデータベース 1 9 0 にアクセスを行

う。

図 2 に本発明を適用したプログラム生成手順を示す。画面ファイル 2 0 0 内に業務プログラムを作成できるように、業務プログラムの関数名、言語、パラメータのデータ定義、また業務プログラムを呼出するための呼出し部につける名前等の情報を定義情報 2 0 1 に定義する。定義情報 2 0 1 を生成ツール 2 1 0 が読み込んで画面から業務プログラムを呼出するために必要なファイルを生成する。生成物それぞれの役割を説明する。動的データ参照画面ファイル 2 2 0 は、画面からの入力や、業務プログラムからデータを取得して表示させるといった、動的にデータの入出力を行う機能を持つ。プログラム呼出し部のソースファイル 2 3 0 及び Native インタフェースのソースファイル 2 4 0 は画面と業務プログラムのデータの入出力を仲介をする役割を持ち、データ型の変換やデータ整形、パラメータの抽出等の機能を持つ。業務プログラムの雛型 2 5 0 は定義された関数名、画面と連携する部分のデータ定義情報をあらかじめ設定しているので、開発者は、処理の実装部分だけコーディングすればよい。

#### 【 0 0 1 3 】

生成ツールで生成されたファイルを、それぞれマシンに依存したコンパイラ 2 6 0 を用いてコンパイルする。それぞれコンパイルしたものが、プログラム呼出し部 2 7 0、Native インタフェース 2 8 0、業務プログラム 2 9 0 になる。

#### 【 0 0 1 4 】

図 3 以降に図 2 に示した生成手順の中に出てきたファイルやプログラムについて、実施例を示しながら説明していく。

図 3 に画面ファイルの表示例を示す。この画面 3 0 0 は、名前を入力させるためのテキストフィールド 3 1 0、パスワードを入力させるためのテキストフィールド 3 2 0、フォームデータを送信するための送信ボタン 3 3 0 が配置されている。

#### 【 0 0 1 5 】

図 4 に画面ファイル 2 0 0 の詳細コードを示す。本例は画面を表示制御するタグ言語で記述されている。定義情報 2 0 1 で、jfd という独自の拡張タグを用いて、本発明独自の情報を定義している。下線部 2 1 2 で関数情報を定義する。cla



ssName属性にプログラム呼出し部の名称を指定している。dllName属性、program Name属性には呼出す業務プログラムのDLLファイル名と実際のプログラム名（関数名）を指定している。Word属性で言語を指定している。scope属性、id属性は動的データ参照画面ファイルのライフサイクルの範囲指定と、プログラム呼出し部にアクセスするための識別子を定義している。下線部 2 1 3 でデータ定義情報を定義する。name属性で指定する名称を画面ファイル内の入出力オブジェクトの 1 つと同一名称にすることで、オブジェクトとの関連付けが行える。interface属性では、業務プログラム内で使用するデータの名称を指定する。size属性ではデータ長を指定する。type属性ではデータ型を指定する。alignment属性にはけた寄せを指定する。paddingChar属性には埋め字の埋める文字（スペース、ゼロ、NULL文字等）を指定する。

#### 【 0 0 1 6 】

図 5 は生成の段階で、拡張タグによって定義された情報を抜き出してテーブルにしたものである。定義情報 5 0 0 内の関数情報 5 0 1 には画面に対応したプログラム呼出し部の名称がSample.Pageであり、呼出す業務プログラムのDLL名称がSAMPLE、プログラム名がNEWUSERという名称で、開発言語はCOBOLであるということを示している。データ定義情報 5 0 2 では画面ファイル 2 0 0 にnameという名称で定義されたデータが、業務プログラムではUSERNAMEという変数にマッピングし、データ型はkanji型で、データ長は 1 0 文字、埋め字にはスペースを使用するという情報を示している。idという名称で定義されたデータは、業務プログラムではUSERIDという変数にマッピングし、データ型はcharacter型で、データ長は 8 文字、埋め字にはzero（“0”）を使用するという情報を示している。buttonという名称で定義されたデータは、業務プログラムではSYORIFLUGという変数にマッピングし、データ型はnumber型で、データ長は 1 文字、埋め字は使用しないという情報を示している。

#### 【 0 0 1 7 】

図 6 は生成ツール 2 1 0 の処理フローを示している。図 6 の説明は図 4、図 5、図 7、図 8、図 9 を参照しながら行う。図 4 に示す画面ファイル 2 0 0 の定義情報 2 0 1 を読取る処理 6 1 0 で、画面ファイル 2 0 0 で拡張タグであるjfdで

指定された項目を抜き出し、定義されている属性値から読取った情報で図5に示す定義情報テーブル500を作成する。定義情報テーブル500内には、関数情報テーブル501とデータ定義情報テーブル502がある。

#### 【0018】

関数情報テーブル501の呼出し部名称とは画面に対応した呼出し部の名称を表して図4の下線部212のclassNameの属性値である。パッケージ名称は業務プログラムをdllファイルにした場合のdllのファイル名称を指している。Jfdタグ内のdllNameの属性値である。関数名は呼出す業務プログラムの名称を表していて、programNameの属性値である。言語は業務プログラムの開発言語を指定していて、Wordの属性値である。

#### 【0019】

データ定義情報テーブル502の名称は画面ファイル200において、実際のフォームオブジェクトに関連付けた名称になっていて図4の下線部203のnameの属性値である。COBOL名称は業務プログラム内で、使用されるデータ定義名で、画面から入力されるデータ名を業務プログラム用に別の名前をつけたものである。Jfd:dataタグ内のinterfaceの属性値である。データ型は業務プログラムで使用するデータの型を表していてtypeの属性値である。長さはデータ長を表していてsizeの属性値である。整形はデータに必要となる整形（右寄せ、左寄せなど）を表したもので、alignmentの属性値である。埋め字はデータの余った部分に何かしらの文字を詰める処理のことを指し、そのつめる文字が何であるかを指定することで、paddingCharの属性値である。

#### 【0020】

業務プログラム雛型250の言語設定処理620は関数情報テーブル501の言語の項目を元に設定される。業務プログラム雛型250及びNativeインタフェース（ソース）240に呼出す関数名を設定する処理630では、関数情報テーブル501の関数名の項目が参照される。図9の業務プログラムの雛型250のPROGRAM-ID. NEWUSER. が設定される。データ定義情報検出分岐処理640では、データ定義情報テーブル502から1レコードずつ読み込んでいく。データが定義されている場合はデータ型の項目を参照してデータ型変換メソッドの生成と長

さ、整形及び埋め字の項目を参照してデータ整形メソッド生成の処理 6 5 0 を行う。図 8 に示すプログラム呼出し部（ソース） 2 3 0 内のデータ変換部 2 3 1 に jfdConvJStr メソッドと jfdLjust, jfdRjust メソッドが生成される。データ取得メソッド生成処理 6 6 0 では、画面ファイル 2 0 0 の中で定義されているオブジェクトの name 属性と jfd の拡張タグで定義されているデータの name 属性を照らし合わせて、一致した場合は、そこにプログラム呼出し部からデータを取得するためのメソッドを組み込む処理を行う。図 7 の動的データ参照画面ファイル 2 2 0 内の下線部 2 2 2 にそれぞれ value 属性値に取得メソッドが追加されている。

### 【 0 0 2 1 】

次に業務プログラム雛型 2 5 0 及びプログラム呼出し部 2 3 0 にデータ定義とデータ設定メソッドを設定する処理 6 7 0 を行う。図 8 のデータ変換部 2 3 1 及びデータ抽出部 2 3 2 にそれぞれのデータ定義に対応して生成される。また図 9 の業務プログラムの雛型 2 5 0 内にも LINKAGE SECTION の COPY 句の内容として、対応したデータの定義が生成される。終了判定分岐処理 6 8 0 で、データ定義が 1 つもない場合は、エラー処理 6 8 2 を行いファイルは何も生成しないで終了する。データ定義が 1 つ以上ある場合は、それぞれのファイルの生成処理 6 8 1 を行い終了する。

### 【 0 0 2 2 】

図 7 に動的データ参照画面ファイル 2 2 0 の詳細コードを示す。下線部 2 2 1 ではプログラム呼出し部にアクセスするための設をしている。class の属性値は画面ファイル 2 0 0 の下線部 2 0 2 の className の属性値である。id の属性値は下線部 2 0 2 の id の属性値である。scope の属性値は下線部 2 0 2 の scope の属性値である。画面ファイル 2 0 0 のなかで jfd 拡張タグで定義されている情報は全て削除する。下線部 2 2 2 はフォームオブジェクトの記述部分だが、これは画面ファイル 2 0 0 内で下線部 2 0 3 のデータ定義情報の中に同じ name の属性値がある場合に、そのデータを取得するためのメソッドを value 属性に指定している。こうすることで、業務アプリケーションから、何らかのデータを取得して初期値として表示することが可能になる。

### 【 0 0 2 3 】

図 8 にプログラム呼出し部 2 3 0 の詳細コードを示す。データ型の変換、整形処理部 2 3 1 では、Native インタフェースにデータを設定する setName, setId, setButton というメソッドが、画面から入力されるデータに対応して定義されている。それぞれのメソッドは内部で setUsername, setUserid, setSyoriflug というメソッドを呼出し、実際に業務プログラムにデータを設定する。パラメータとして指定しているデータは jfdConvJStr というメソッドに、文字列化したデータそのものと埋め字用の文字（スペース、ゼロ等）と文字列の長さを表す数字を渡すことで、返り値として指定した長さに、余白部分を指定した埋め文字で埋められた文字列を得る。そのデータを jfdDataRjust または jfdDataLjust というメソッドに渡すことで、右寄せ、または左寄せされた文字列データを取得し、そのデータを業務プログラムに設定している。

#### 【 0 0 2 4 】

パラメータ抽出及び設定処理部 2 3 2 では getParameter メソッドを使ってリクエストの中から name, id, button という名称で設定されているパラメータをそれぞれ取得する。また抽出したデータはデータ型の変換、整形処理部 2 3 1 で定義された設定メソッドを使用して Native インタフェースに設定している。

#### 【 0 0 2 5 】

データの抽出とデータの変換、設定処理を全て終わらせてからプログラム呼出し部 2 3 3 で callCOBOL メソッドで業務プログラム呼出し処理を行っている。

#### 【 0 0 2 6 】

図 9 に業務プログラム雛型 2 5 0 の詳細コードを示す。業務プログラム雛型 2 5 0 は画面ファイル 2 0 0 内の定義情報 2 0 1 を元に関数名や画面との入出力を行うためのデータ定義が既に記述されている部分と開発者が独自に実装するデータ定義部 2 5 1 と実処理部 2 5 2 からなる。開発では生成された業務プログラム雛型のファイルに開発者が実処理部 2 5 2 をコーディングしてからコンパイルを行い配置することになる。「IDENTIFICATION DIVISION.」、「PROGRAM-ID.」、「DATA DIVISION.」、「WORKING-STORAGE SECTION.」、「LINKAGE SECTION.」、「PROCEDURE DIVISION USING GYOMU-A.」、「EXIT-PROGRAM」および「END-」の部分については、COBOL 言語のプログラムを生成するためのテンプレ

ートとして記憶している部分であり、業務プログラムの生成時に自動生成する。前記抽出したパラメータから「01 GYOMU-A.」、「02 USERNAME PIC N(10).」、「02 USERID PIC X(8).」および「02 SYORIFLUG PIC X(1).」を生成する。構造体名である「GYOMU-A」は、システムで自動的に割り付けることも可能であるし、操作者からの入力を受けて設定することも可能である。

#### 【0027】

図10にコンパイルされたプログラム呼出し部270の詳細図を示す。図3に示す画面300から名前用のテキストフィールド310、パスワード用のテキストフィールド320及びボタン330が図7に示すようにそれぞれname、id、buttonというパラメータ名でリクエスト1000内に格納されて送信されてくる。プログラム呼出し部270はリクエスト1000を受け取り、パラメータ抽出部271でnameパラメータに格納されているデータ、idパラメータに格納されているデータ、buttonパラメータに格納されたデータをそれぞれ抽出する。業務プログラムに適した型や、形に変換するデータ変換部273で抽出されたデータは図4の画面ファイルに示されているように、nameは業務プログラムのUSERNAMEという変数にマッピングし、kanji型でデータ長は10文字、埋め字はスペースとなるように変換される。idは業務プログラムのUSERIDという変数にマッピングし、character型でデータ長は8文字、右寄せで、埋め字は"0"となるように変換される。buttonは業務プログラムのSYORIFLUGという変数にマッピングし、number型でデータ長は1文字となるように変換される。

#### 【0028】

プログラム呼出し部272は図8に示すように抽出したデータをデータ変換し、その結果業務プログラムに必要なパラメータ1010をNativeインタフェースに渡す。更に必要なパラメータを全てを設定した後に業務プログラムを実行させる。

#### 【0029】

図11にデータ変換部の処理フローを示す。言語分岐1100、Java用型変換処理1110、Java用けた寄せ処理1111、Java用埋め字処理1112、COBOL用型変換処理1120、COBOL用けた寄せ処理1121、COBOL用埋め字処理1

1 2 2、C言語用型変換処理 1 1 3 0、C言語用けた寄せ処理 1 1 3 1、C言語用埋め字処理 1 1 3 2 からなる。

図 1 2 にけた寄せと埋め字について示す。記号は説明 1 2 0 0 を参照する。何もしない場合は 1 2 1 0 の入力に対して、1 2 1 1 のマッピングになる。右にけた寄せを行うと 1 2 2 0 の入力に対して 1 2 2 1 のように格納される。左にけた寄せを行い、埋め字スペースを用いると、1 2 3 0 は 1 2 3 1 のように格納される。右寄せ、埋め字がスペースだと 1 2 4 0 は 1 2 4 1 のように格納される。数値に対し、何も行わないと 1 2 5 0 は 1 2 5 1 のように格納される。数値に対し、右寄せを行い埋め字半角スペースを用いると 1 2 6 0 は 1 2 6 1 のように格納される。右寄せ、0 を埋め字に用いると 1 2 7 0 は 1 2 7 1 のように格納される。

### 【0 0 3 0】

図 1 3 に本発明を用いた開発環境のエディタ 1 3 0 0 の概念図を示す。画面実際に表示させる画面ビュー 1 3 1 0 を見ながら、画面ファイルをコードビュー 1 3 2 0 から編集を行う。また、画面ファイルの定義情報から生成される業務プログラムの雛型が業務プログラムビュー 1 3 3 0 に表示されるので、開発者は、開発者が実装する処理部 1 3 3 1 を編集し、業務プログラムを作成する。生成物の全体の流れを説明する。

### 【0 0 3 1】

動的データ参照画面ファイル 2 2 0 で表示される画面 3 0 0 のテキストフィールド 3 1 0 に名前を入力しテキストフィールド 3 2 0 にパスワードを入力したあと送信ボタン 3 3 0 を押下すると名前はname、パスワードはid、送信ボタンを押下した情報はbuttonというパラメータとして設定され、パラメータはその他の送信に必要なデータとともに連結されて、リクエスト 1 0 0 0 という形になり図 1 に示すネットワーク 1 4 0 を介してWebサーバ 1 5 0 上にあるServlet 1 6 0 に送信される。Servlet 1 6 0 ではリクエスト 1 0 0 0 を受け取ると画面に対応したプログラム呼出し部にリクエスト 1 0 0 0 をそのまま転送する。

### 【0 0 3 2】

プログラム呼出し部 2 7 2 はパラメータ抽出部 2 7 1 でリクエスト 1 0 0 0 を受け取るとgetParameterメソッドで、name、id、button、というパラメータをそ

れぞれ抽出し、データ変換部 2 7 3 でそれぞれのデータをnameはkanji型、1 0 文字、スペースで埋め字処理、idはcharacter型で 8 文字、右寄せで格納し” 0 ”で埋め字処理、buttonはnumber型で 1 文字というように変換を行う。それぞれのパラメータは業務プログラムでnameはUSERNAME、idはUSERID、buttonはSYORIFLUGと関連づいているので、Nativeインタフェース (COBOL) 1 7 0 の設定メソッドを使用して、パラメータを構築する。パラメータの構築が終わり次第、プログラム呼出し部 2 7 2 がCOBOLの業務プログラムを実行する。図 9 の業務プログラム雛型 2 5 0 内の開発者実装部分 2 5 1 及び 2 5 2 をコーディングしてコンパイルした、業務プログラム 2 9 0 は実行すると、あらかじめNativeインタフェース 2 8 0 が業務プログラム 2 9 0 からパラメータが参照できるように設定しているので、構築されたパラメータを参照しUSERNAME、USERID、SYORIFLUGを使用した処理を行う。

### 【 0 0 3 3 】

#### 【発明の効果】

本発明によれば、種々のプログラム言語でコーディングされた業務プログラムを容易に連携することが可能となる。

#### 【図面の簡単な説明】

【図 1】 本発明の生成物の構成の実施例である。

【図 2】 本発明のプログラム生成の概念図である。

【図 3】 本発明の画面ファイルの表示例である。

【図 4】 本発明の画面ファイルの詳細コードである。

【図 5】 本発明の定義情報のテーブルである。

【図 6】 本発明の生成ツールの処理フローである。

【図 7】 本発明の動的データ参照画面ファイルの詳細コードである。

【図 8】 本発明のプログラム呼出し部の詳細コードである。

【図 9】 本発明の業務プログラム雛型の詳細コードである。

【図 1 0】 本発明のプログラム呼出し部の概念図である。

【図 1 1】 本発明のデータ変換の処理フローである。

【図 1 2】 本発明のデータ変換の実施例である。

【図 1 3】 本発明の開発環境エディタの概念図である。

【符号の説明】

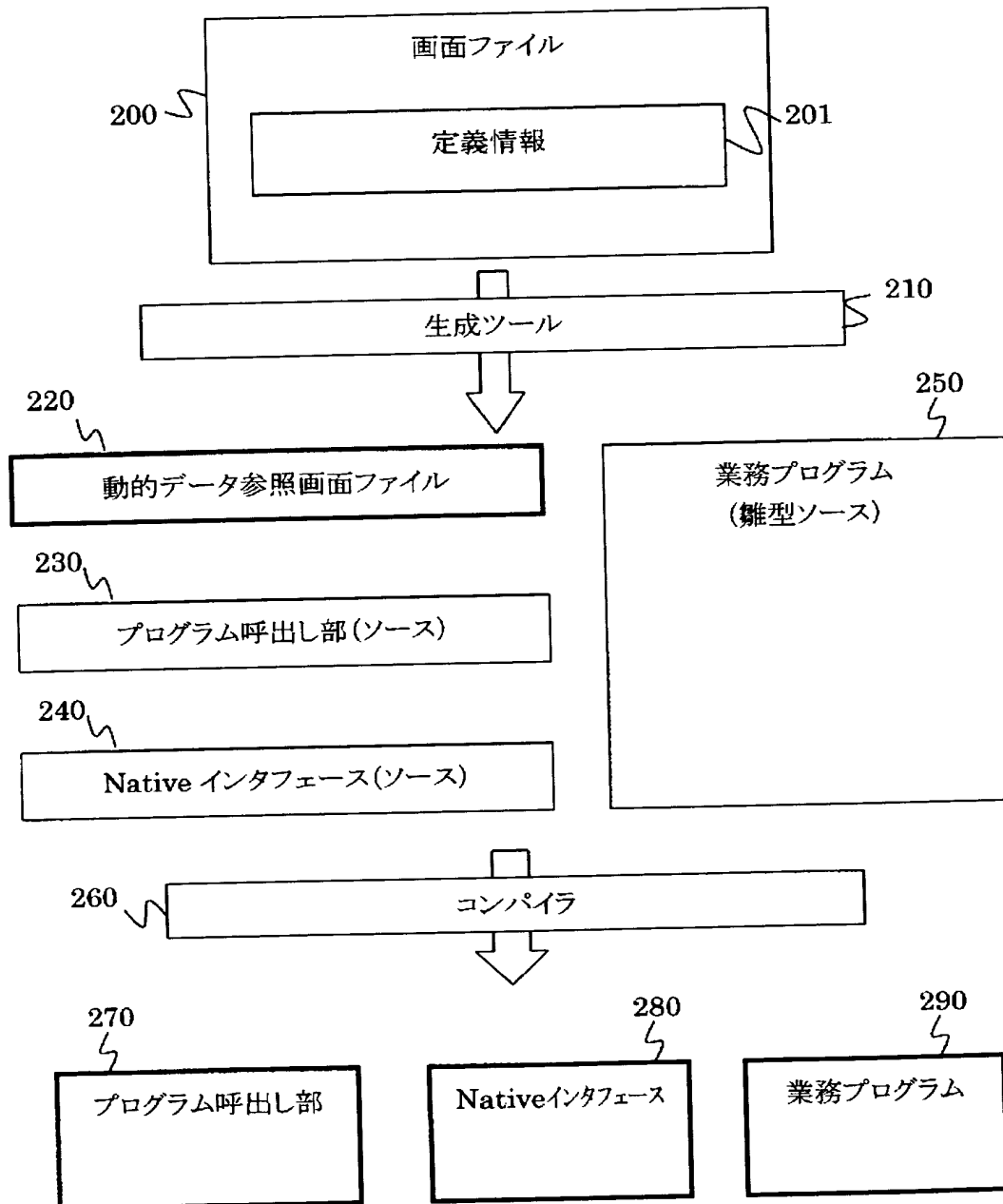
- 1 0 0 端末
- 1 1 0 画面 A
- 1 1 1 プログラム呼出し部 A
- 1 1 2 業務プログラム A (COBOL)
- 1 2 0 画面 B
- 1 2 1 プログラム呼出し部 B
- 1 2 2 業務プログラム B (COBOL)
- 1 3 0 画面 C
- 1 3 1 プログラム呼出し部 C
- 1 3 2 業務プログラム A (Java)
- 1 4 0 ネットワーク
- 1 5 0 Webサーバ
- 1 6 0 Servlet
- 1 7 0 Nativeインタフェース (COBOL)
- 1 8 0 Nativeインタフェース (Java)
- 1 9 0 データベース





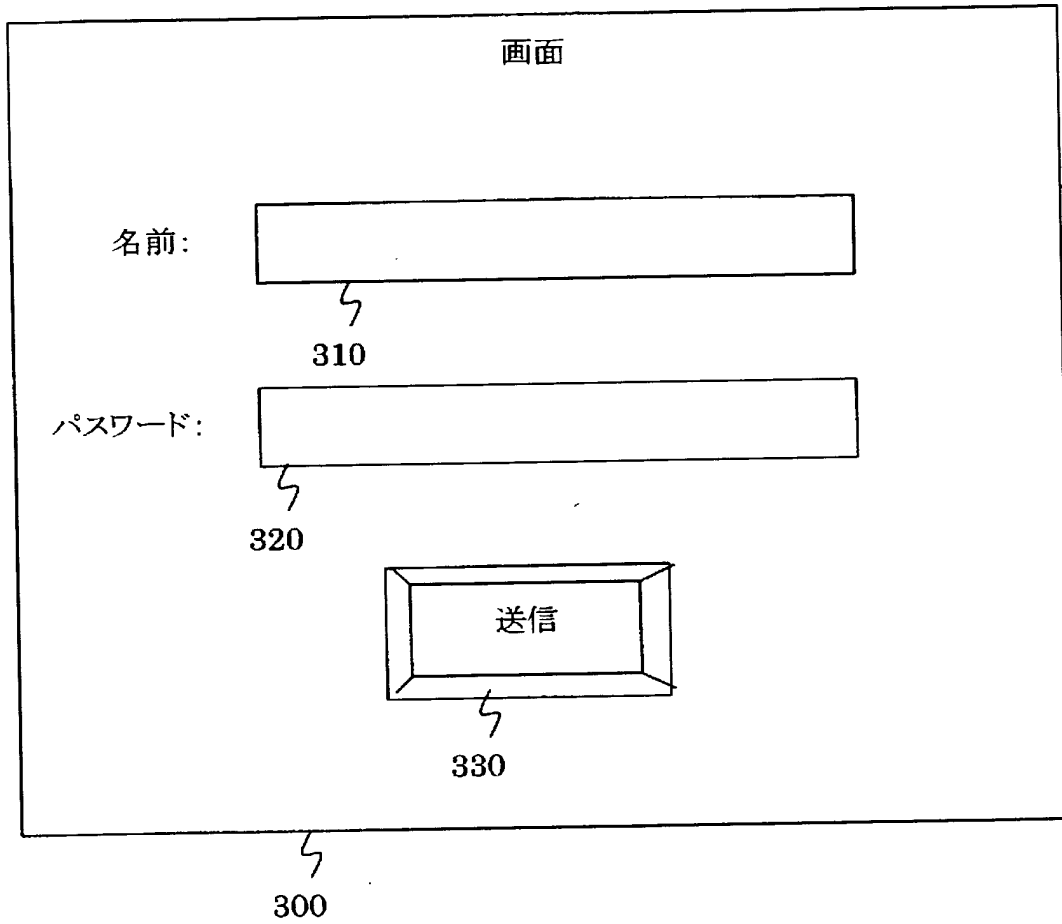
【図 2】

図2



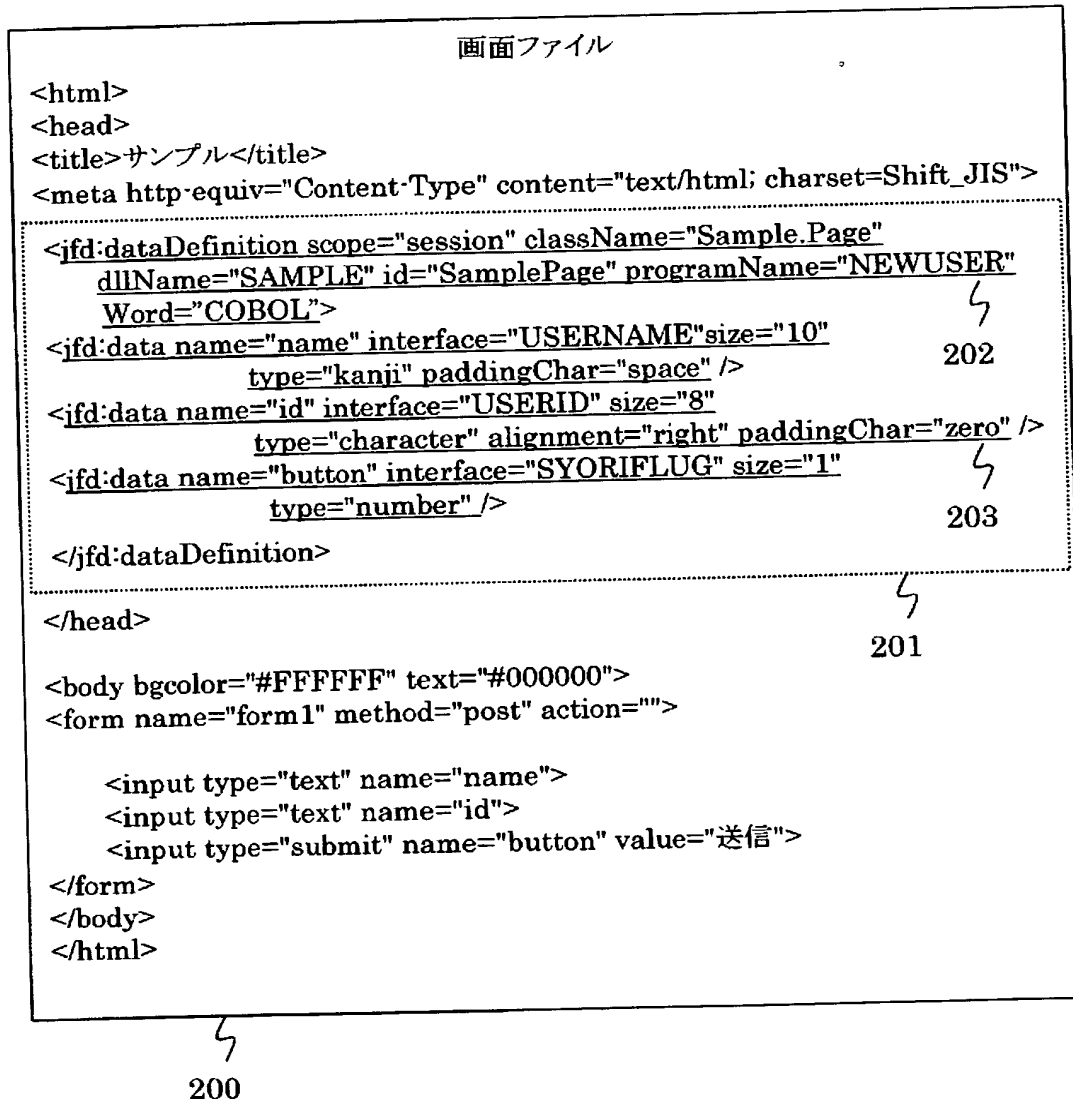
【図3】

図3



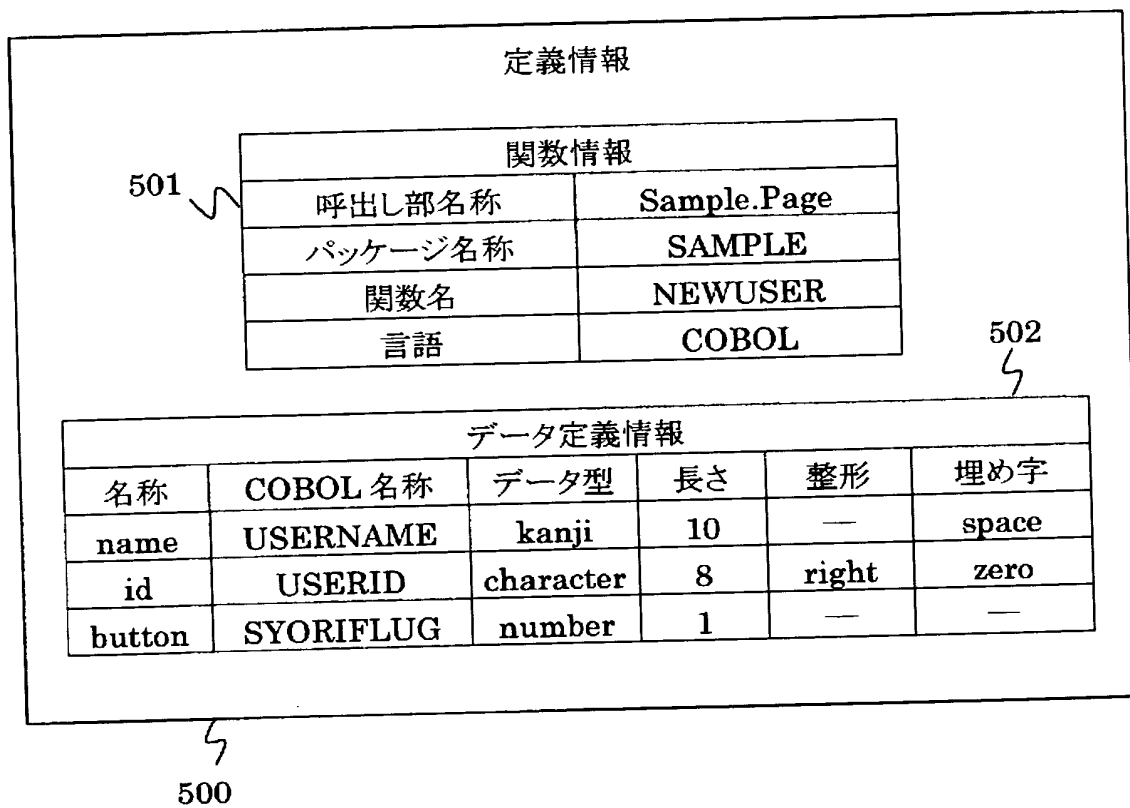
【図 4】

図4



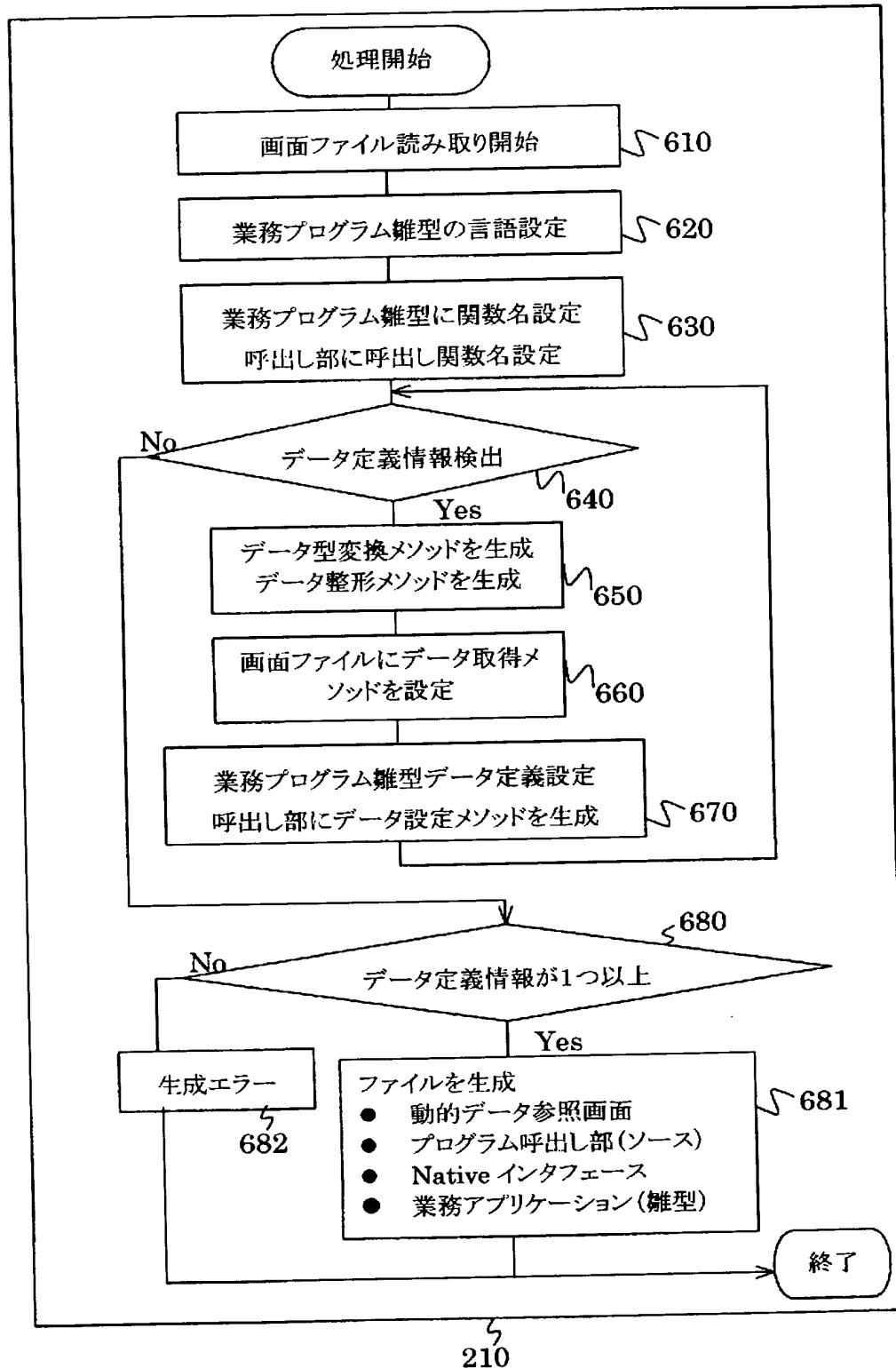
【図 5】

図5



【図6】

図6



【図 7】

図 7

動的データ参照画面ファイル

```

<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page import="java.io.File" %>
<jsp:useBean class="Sample.Page" id="SamplePage" scope="session" />
<html>
<head>
<title>サンプル</title>
<meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
</head>

<body bgcolor="#FFFFFF" text="#000000">

<form name="form1" method="post" action="Sample">
  <input type="text" name="name" value="<%= SamplePage.getName_value() %>">
  <input type="text" name="id" value="<%= SamplePage.getId_value() %>">
  <input type="submit" name="button"
    value="<%= SamplePage.getButton_value() %>">
</form>

</body>
</html>

```

220

【図 8】

図8

プログラム呼出し部 (ソース)

```

package Sample;

public class Page {
    Page() {}

    public void setName(Object data) {
        myBean.setUsername(jfdDataLJust(jfdConvJStr(data.toString()), " ", 20));
    }
    public void setId(Object data) {
        myBean.setUserid(jfdDataRJust(jfdConvJStr(data.toString()), "0", 8));
    }
    public void setButton(Object data) {
        myBean.setSyoriflug(jfdDataLJust(jfdConvJStr(data.toString()), " ", 1));
    }

    public Object getName() {}
    public Object getId() {}
    public Object getButton() {}

    public void execute(HttpServletRequest req) {
        String name = req.getParameter("name");
        setName(name);
        String id = req.getParameter("id");
        setId(id);
        String button = req.getParameter("button");
        setButton(button);

        myBean.callCOBOL();

    }
    
```

230



【図 9】

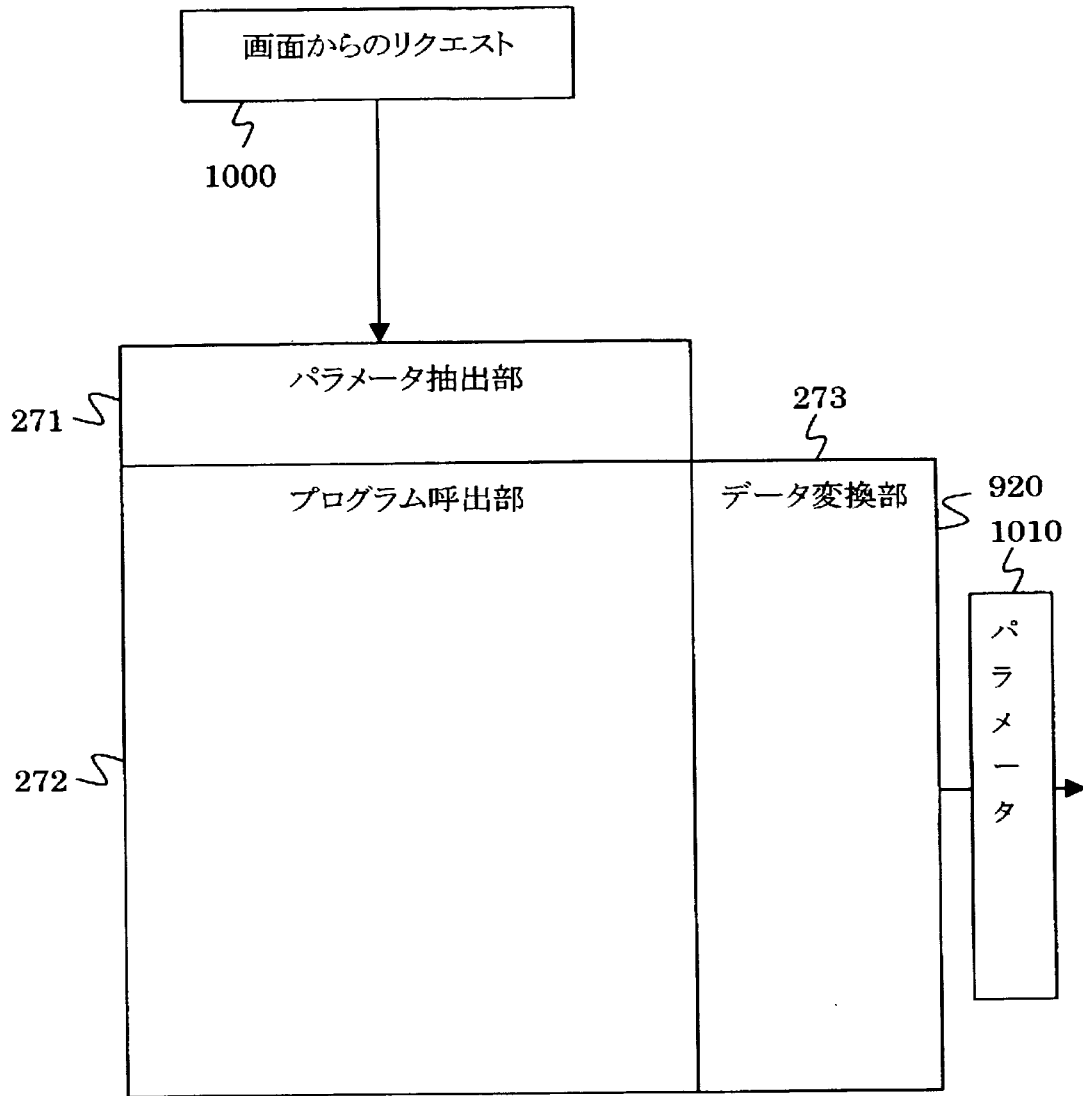
図 9

業務プログラム (COBOL)		
IDENTIFICATION DIVISION.		
PROGRAM-ID. NEWUSER.		
DATA DIVISION.		
WORKING-STORAGE SECTION.		
251 ⚡		
01	SYSTEMNAME	PIC N(10).
01	SYSTEMID	PIC X(8).
LINKAGE SECTION.		
01	GYOMU-A.	
02	USERNAME	PIC N(10).
02	USERID	PIC X(8).
02	SYORIFLUG	PIC X(1).
PROCEDURE DIVISION USING GYOMU-A.		
252 ⚡		
	MOVE USERNAME	TO SYSTEMNAME
	MOVE USERID	TO SYSTEMID
	MOVE 0	TO SYORIFLUG
EXIT-PROGRAM		
END-NEWUSER.		

⚡  
250

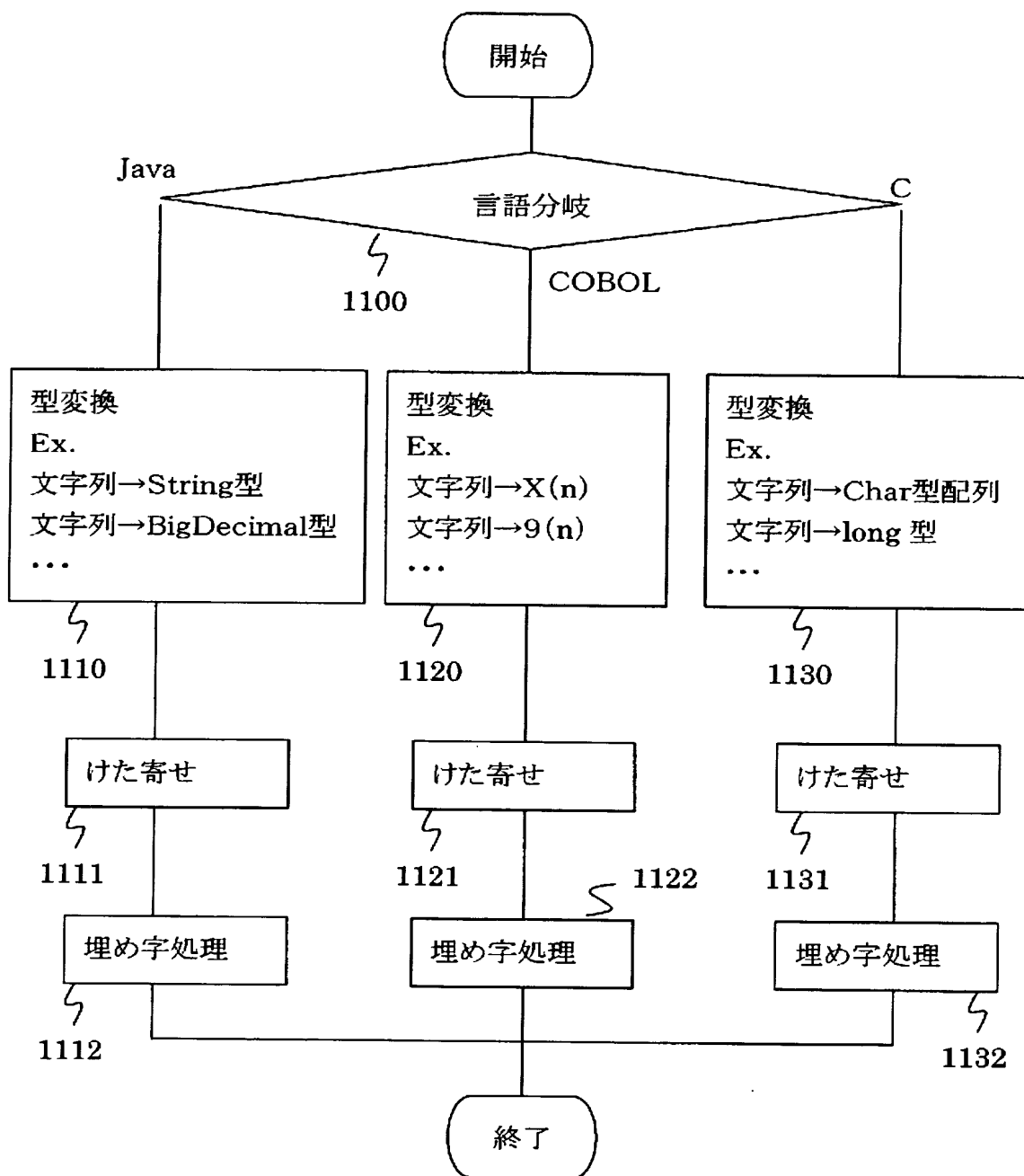
【図10】

図10



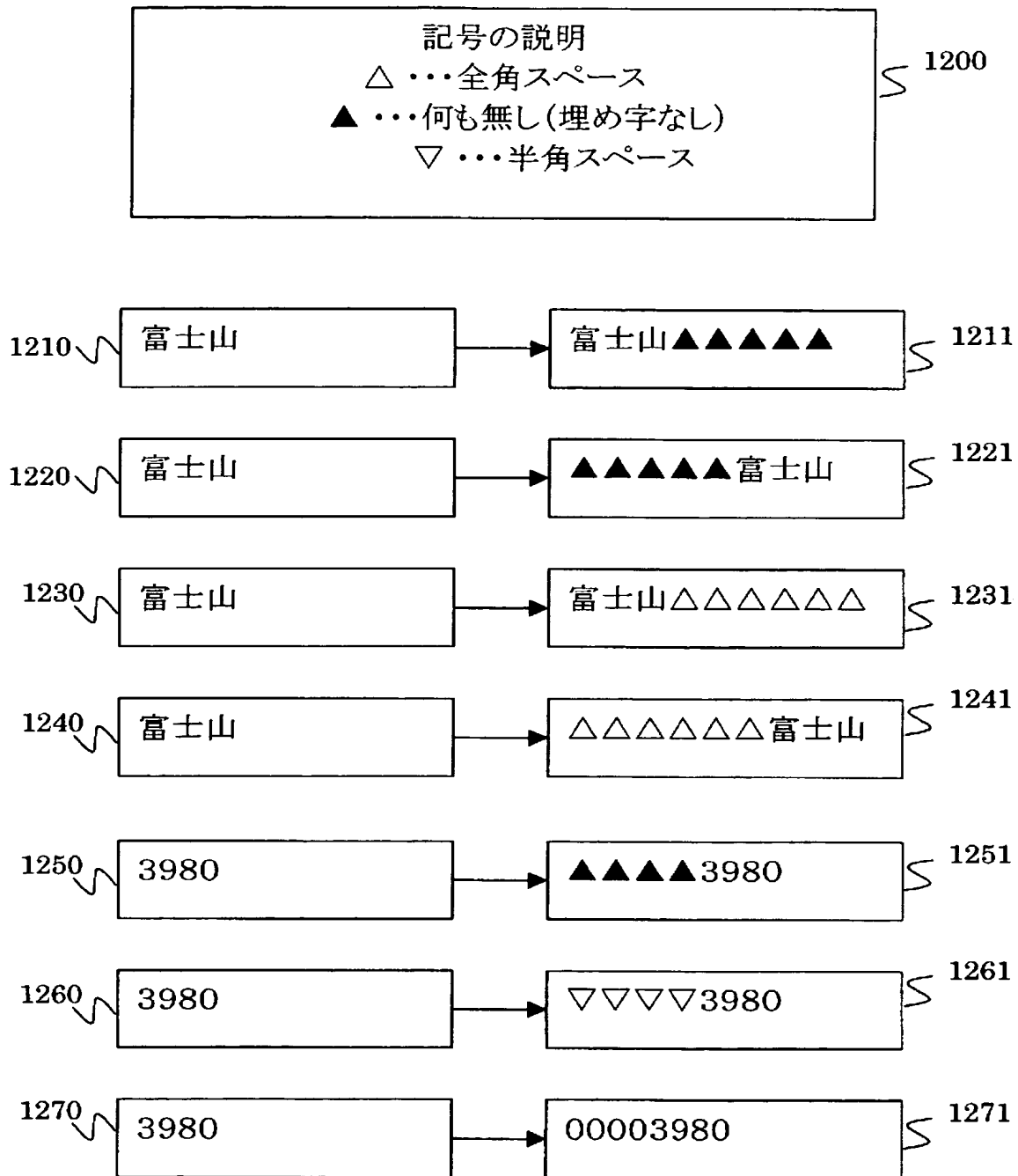
【図 11】

図11



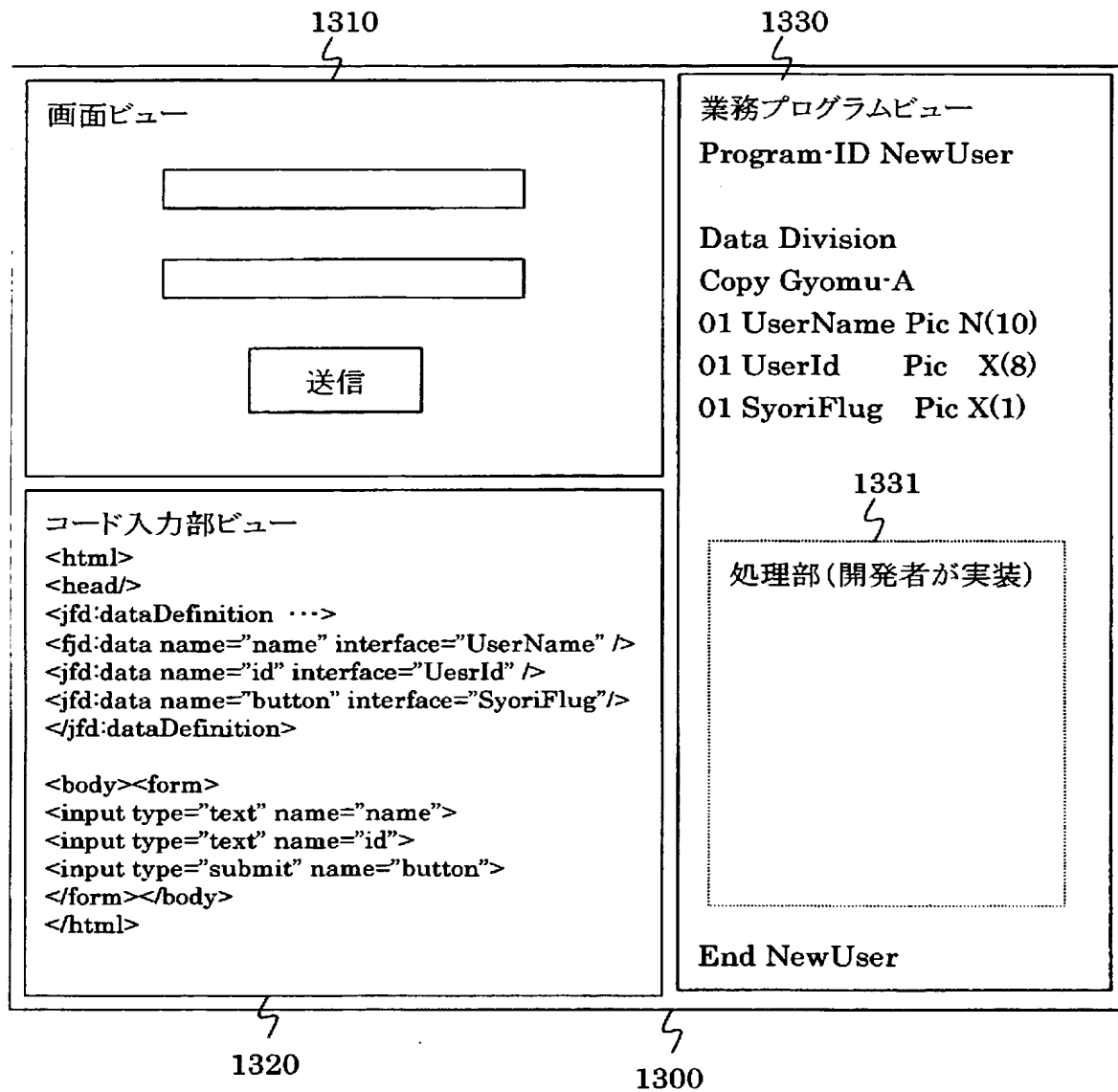
【図 12】

図12



【図13】

図13



【書類名】 要約書

【要約】

【課題】 画面ファイルからの情報を用いて、バックエンドのCOBOLのような言語でコーディングされたプログラムと連携する為の、業務プログラムの雛型を作成する。

【解決手段】 画面ファイルに必要となるフォームの入力データ定義を行い、またバックエンドの業務プログラムの関数名、開発言語を定義させることで、型変換、パラメータの設定を隠蔽する装置を生成させ、それを実装する業務プログラムの雛型を生成する。

【選択図】 図 1

認定・付加情報

特許出願の番号	特願 2 0 0 2 - 1 8 8 9 3 8
受付番号	5 0 2 0 0 9 4 7 3 4 1
書類名	特許願
担当官	第七担当上席 0 0 9 6
作成日	平成 1 4 年 7 月 1 日

< 認定情報・付加情報 >

【提出日】 平成14年 6月28日

次頁無

特願 2 0 0 2 - 1 8 8 9 3 8

出 願 人 履 歴 情 報

識別番号

[ 0 0 0 0 0 5 1 0 8 ]

1. 変更年月日

1 9 9 0 年 8 月 3 1 日

[変更理由]

新規登録

住 所

東京都千代田区神田駿河台 4 丁目 6 番地

氏 名

株式会社日立製作所